Adaptive Infrared Non-Uniformity Correction

February 1999

Robert W. Means and Brad Von Tersch
HNC Software
5930 Cornerstone Court West
San Diego, CA 92121

Dean A. Scribner
Naval Research Laboratories
Code 5636
Washington, DC 20375

ABSTRACT

Real-time adaptive non-uniformity correction by a neural network algorithm was implemented on the 12-bit digital image from a Boeing SE-U20 uncooled 320x240 microbolometer camera. Nonlinearities in an infrared sensor require either periodic recalibration of a one or two point correction algorithm as the scene and environment change or require an adaptive continuous correction. The adaptive neural network correction is performed in real-time with an off-the-shelf processor board inserted in an IBM PC compatible machine. The real time implementation allows long term stability and performance issues of the algorithm to be addressed. Evaluation of the adaptive algorithm shows that the spatial noise in the corrected image depends strongly on the estimate of the desired image used in the adaptive algorithm. The desired value is calculated by means of neighborhood functions such as median or convolution with kernels such as the sinc function. We have determined that the adaptive algorithm works better when the time sample between images of a moving scene is large; that is, when the images are relatively uncorrelated. This effect must be balanced by the need to have the algorithm converge in a finite time. The net effect of this balance is that the hardware signal processing requirements are reduced considerably since the algorithmic calculations need not be done on every frame.

1.0 INTRODUCTION

Infrared image sensors have significant advantages over visible light sensors in low light level surveillance situations and in differentiating and detecting objects that radiate in the infrared, such as engine exhausts, missile launch burns, vehicle engines and human personnel. The disadvantages of infrared sensors include the low temperature refrigeration requirements, the non-uniformity of the sensor response, low signal to noise ratio and high cost. Non-uniformities or defects in the process of manufacturing the sensor often create spatial regions in the sensor with different dark currents and different sensitivity to incident radiation. The refrigeration requirement is tied to the non-uniformity because the dark current rises with temperature. The sensor essentially has a biased noisy signal that depends on the temperature. That is, pixels have values offset from the average due to the dark current non-uniformity or can even be saturated in the presence of no external infrared radiation. Non-uniformities also create regions with different quantum efficiencies and differences in transistor gains in

Form SF298 Citation Data

Report Date *("DD MON YYYY")* 00021999	**Report Type** N/A	**Dates Covered (from... to)** *("DD MON YYYY")*

Title and Subtitle Adaptive Infrared Non-Uniformity Correction	**Contract or Grant Number**
	Program Element Number
Authors	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) HNC Software 5930 Cornerstone Court West San Diego, CA 92121 Monterey, CA 93943 Fort Belvoir, VA 22060	**Performing Organization Number(s)**
Sponsoring/Monitoring Agency Name(s) and Address(es)	**Monitoring Agency Acronym**
	Monitoring Agency Report Number(s)
Distribution/Availability Statement Approved for public release, distribution unlimited	
Supplementary Notes	
Abstract	
Subject Terms	
Document Classification unclassified	**Classification of SF298** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** unlimited
Number of Pages 15	

readout circuitry. This creates a non-uniform gain effect so that even with no non-uniformity in the dark current, a uniformly illuminated sensor will result in a mottled, striped, or otherwise non-uniform image.

The non-uniformities found in sensor arrays can vary both from array to array and within a given array from pixel to pixel. A two-point non-uniformity correction assumes that the value of each pixel can be corrected by multiplying it by a gain and adding an offset to it. This correction is mathematically stated as

$$y_{ij} = G_{ij} * x_{ij} + O_{ij} \quad (1)$$

where x_{ij} is the uncorrected pixel's output value due to the incident light pattern, y_{ij} is the corrected value, and G_{ij} and O_{ij} are the gain and offset coefficients for each pixel. In general, each pixel will have a different value of its gain and offset coefficient. However, if an array has spatially uniform sensitivity, then all the gain coefficients have the same value and all the offset coefficients have the same value. In that case, the correction can be performed by one set of gain and offset values per sensor array. It still may be necessary to provide a distinct gain and offset for each sensor array manufactured because the manufacturing process can vary. On the other hand, if the array has internal non-uniformities that cause the response to vary from pixel to pixel within the array, then each pixel must be corrected by its own gain and offset values. This correction is more difficult to implement than the simple case of spatially uniform sensors. In some sensors, the gain may also be nonlinear and dependent on the incident light level and ambient temperature. This is even more difficult to correct.

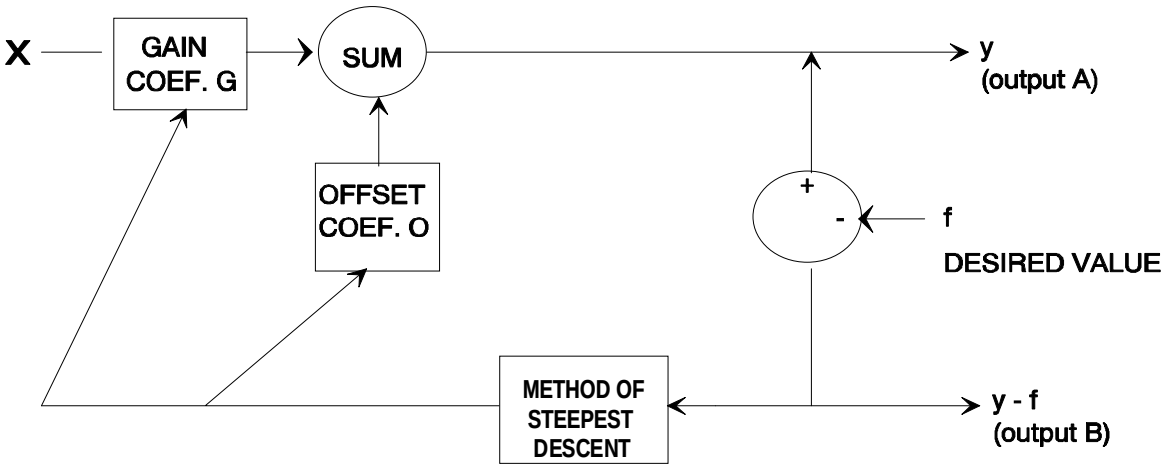
The implementation of a non-uniformity correction must be preceded in some manner by a non-uniformity measurement. If the array is spatially uniform, it is not difficult to test each array's response in the laboratory and provide a single set of gain and offset values for the entire array. A non-uniform sensor array of size 256x256, on the other hand, requires a test and measurement for each of the 65,536 pixels. This is a time consuming and expensive task. Furthermore, these values are only valid for a given ambient temperature and scene temperature. The sensor electronics may even age and the coefficients would be invalid because of that. Thus, camera manufacturers provide users with the capability to periodically recalibrate their cameras.

The offset coefficients of many focal plane sensors used for infrared imaging are temperature dependent. The offsets are caused by dark currents that provide a temperature dependent signal even in the absence of incident radiation. The sensor integration time can also provide a source of variation for the offset. The output offset is directly proportional to the integration over time of the dark current. Thus the corrector, even for a spatially uniform array, may have to implement a complex temperature and frame rate dependent algorithm to calculate the required offset. In addition, the temperature dependence must be measured in the laboratory and the appropriate constants provided for the corrector's algorithm. The gain is less dependent on temperature, but some sensors may require this correction also. These correction methods also require the measurement of temperature in the field. Thus it is desirable to develop an adaptive non-uniformity corrector that does not depend on measurements made in the laboratory, but can make correction to the pixel values in real time based solely on the scene contents.

Most uncooled IR cameras correct the image by imposing an analog offset of limited precision to each pixel before calculating a more precise digital gain and offset for each pixel. For the most part, this paper assumes that the analog correction is already made and that the adaptive correction is being done within the dynamic range of A/D converter after the analog offset is done. . This is discussed more thoroughly in section 3.

2.0 ADAPTIVE NON-UNIFORMITY CORRECTION

The adaptive neural network algorithm, that was proposed and tested by Scribner et al [1], is essentially a recurrent neural network based on a Least Mean Squares algorithm similar to those developed by Widrow [2]. It optimizes a set of gain and offset values for every pixel in the image. Figure 1 is a simplified flow chart of the algorithm. The desired image, *f*, is created from the input image, *x*, by a local neighborhood interpolation. It is then compared to the gain and offset corrected image, *y*, to generate the error term, *(y – f)*. That term is then used to adjust the gain and offset coefficients to eliminate the error in a least mean square sense.



$$O_{n+1} = O_n - 2\alpha(y - f)$$
$$G_{n+1} = G_n - 2\alpha x(y - f)$$

Figure 1. Neural net algorithm for adaptive non-uniformity correction

The spatial interpolation operation over a neighborhood, as described above, is the mathematical operation of convolution with a kernel whose values are given by the sinc function. Neighborhood sizes (or equivalently, convolution kernel sizes) from 3x3 to 21x21 were tested by Scribner et al [1]. The equations in Figure 1 are derived by the method of steepest descent and can be translated into a more detailed flow-chart block diagram representation as illustrated in Figure 2.

The block diagram provides insight as to how the algorithm can be implemented in hardware and software. Each one of the functions called out in the block diagram (convolution, multiplication, subtraction, and addition) is a basic operation that the our hardware performs through a software library function call. Each of the blocks in the algorithm is implemented sequentially. The image from the sensor is used as an input for several of the operations. Thus, it must first be read out of the sensor and stored in memory so that it is available at each point in the sequential algorithm. From memory, the image is convolved with a kernel to produce an estimate of the desired image. This estimate is “desired” in a statistical sense. The long-term average of the estimate should be equal to the long-term average of the true value of the pixel. The desired image is then subtracted from the corrected image and serves as an error term. The error term is then used to correct the gain and offset. The algorithm proceeds in this manner, executing each block operation sequentially, while storing intermediate results in memory.

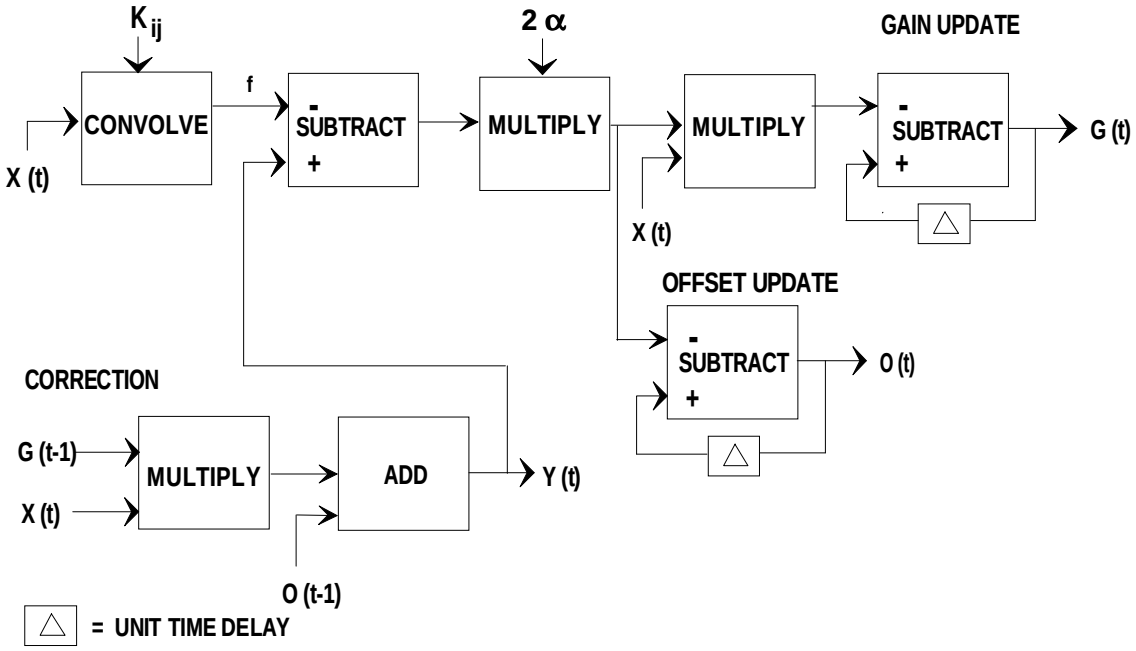


Figure 2. Non-uniformity correction algorithm block diagram

Scribner et al [1] experimented with inserting additional feedback into the calculation of the gain and offset function by using **y(t)** in place of **x(t)** in the upper half of the algorithm illustrated in Figure 2. The use of **y(t)** can lead to instabilities, but if the adaptation rate is small enough the algorithm remains stable. We experimented with using **y(t)** in place of **x(t)** and found no significant difference in image quality when the algorithm was stable. Consequently, none of the images in this paper use **y(t)** in place of **x(t)**.

3.0 REAL-TIME IMPLEMENTATION AND RESULTS

3.1 Processing Hardware

We have used a Matrox Genesis board with a Texas Instruments TMS320C80 chip to perform the image processing. The board is plugged into a PCI slot on a Dell 200 MHz Pentium Pro computer. Matrox provides a library of functions callable from a Microsoft Visual C++ environment that are sufficient to perform the required image processing. The C80 chip has four 32-bit integer processors and one 32-bit floating point processor running in parallel at 50 MHz. The 32-bit integer processors can perform four 8-bit operations per cycle or two 16-bit operations per cycle. The C80 can perform, in total, over 2 Billion 8-bit operations per second. In addition, Matrox provides a separate chip, the Neighborhood Operations Accelerator, for accelerating neighborhood operations such as convolution. It can typically provide a factor of 8 to 20 over the C80 for convolution. Typical processing times for elementary operations on a 16-bit image on size 320x240 are given in Table 1.

Table 1. Typical Processing Times for Elementary Operations

Operation	Processing Time
Add two images	2.07 ms
Multiply two images	2.37 ms
3 x 3 convolution	1.72 ms
3 x 3 median filter	6.80 ms
15 x 15 convolution	12.20 ms
12-bit histogram equalization	13.73 ms
Display 8-bit image	0.87 ms
Store 8-bit image to disk	23.39 ms
Store 16-bit image to disk	39.93 ms

3.2 Algorithm Complexity

All uncooled IR camera manufacturers provide a simple two-point calibration capability for the user to generate fixed gain and offset coefficients. The signal processing is done on board the camera within some very strict power and weight limits. In addition, a dead pixel substitution algorithm replaces known dead pixels with a nearby neighbor. In Boeing's SE-U20, the substituted pixel can be any neighbor in an 8x8 region. The adaptive algorithm, described in section 2, adds significant complexity to the signal processing requirements. This section goes into the details of the algorithm and its implementation.

3.2.1 Arithmetic Precision

The method of steepest descent works by adding small corrections to the gain and offset coefficients and minimizing the resultant error. Thus the coefficient, α, in the equations (2) is usually very small.

$$O_{n+1} = O_n - 2\alpha(y - f)$$
$$G_{n+1} = G_n - 2\alpha x(y - f) \qquad (2)$$

Arithmetic with only 8-bits of precision is inadequate to this task. All images in the equations (2) can be adequately represented by 16 bits. The arithmetic operations, such as subtracting two images are also done adequately with 16 bits. However, the Gain and Offset coefficient arithmetic requires 32-bit precision for storing intermediate products and adding small error terms to large quantities. The combination of different levels of required precision makes the C80 and the corresponding Matrox Genesis software library an excellent choice to implement these algorithms.

Integer arithmetic has a set of concerns and problems that would not occur in floating point arithmetic and one must be very careful to address them in the software. For instance, α cannot be too small or else we would always be adding zero in equations (2). It is convenient to let α be the inverse of a power of 2 so that multiplying by α is equivalent to a right shift. However, the right shift of a small negative number has, because of the automatic sign extension of 2's complement negative numbers, a lower limit of –1. That is, -1 is represented as a 16-bit integer by 1111111111111111. No matter how many times we right shift this with sign extension, it never goes to zero. And thus, there is a systematic bias in the gain and offset calculation unless we take this into account.

3.2.2 Convolution Edge Effects

The desired image is created from the raw image (after dead pixel correction) by a convolution with a fairly large kernel. At the edges of the image, a large part of the kernel is outside of the image as illustrated in Figure 3 for a 7 x 7 kernel. In the convolution calculation, the pixels outside of the image are set equal to zero. This creates systematic errors in the desired image that will result in visible artifacts. One method to avoid these artifacts is to redefine the image of interest to have fewer pixels than the sensor itself. Thus, if the kernel size is 15 x 15 and the sensor is 320 x 240, the image, for which gain an offset coefficients are computed, is 306 x 226. With the smaller image, we have valid values for the pixels outside the 306 x 226 image of interest and the convolution results are valid. The problem with this method is that a large fraction of the image is then uncorrected. A second method is to renormalize the results of convolution after assuming that the pixels outside of the image are zero. The utility of this solution can be best seen by assuming that we have chosen a kernel of all 1's, normalized by the sum of all the values in the kernel. Thus our best estimate of the desired pixel value (for a 15 x 15 kernel) is the sum of the all the pixels in the 15 x 15 pixel neighborhood divided by 225. This works well in the interior of the image, but at the edges, we want to divide by the count of pixels inside the image instead of 225. Convolving an image of all 1's with the kernel of all 1's easily gets this count. The image generated is used to normalize the result of convolving the raw image. We will actually use a kernel with coefficients generated by a circularly symmetric sinc function (sin(x) / x) to provide the desired image. In that case, we will use the result of convolving the sinc function kernel with an image of all 1's as the normalizing image.

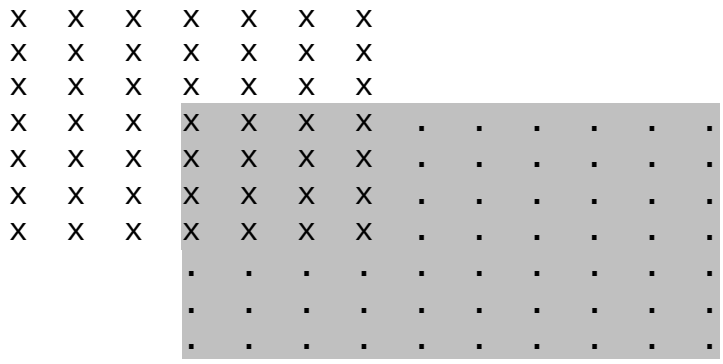


Figure 3. Placement at image edge of initial 7 x 7 convolution kernel for first output result.

3.2.3 Dead Pixels

Some pixels in the sensor are often either zero, saturated or too noisy to use. The Boeing SE-U20 camera replaces, via hardware in real time, those pixels with a pixel chosen from an 8 x 8 surrounding neighborhood. The choice of pixels is done at the factory and is fixed. They also provide the user a map of the dead pixels. By having an 8 x 8 neighborhood choice, very large blobs can be accommodated. After observing that most dead pixels are either point or line defects and that the sensor had no very large blobs of dead pixels, we decided to use a four-corner, 3 x 3 averaging kernel as the estimator for dead pixels. This kernel provides a better estimate of point and line defects than a simple fixed neighbor replacement and is shown in Figure 4.

1 0 1
0 0 0
1 0 1

Figure 4. Dead Pixel Replacement Kernel.

Convolving this kernel with the complement of the dead pixel map (an image that has 0's in dead pixels and 1's in good pixels) provides us with a normalization factor to use in the same manner as that done for the edge effect normalization. A single pass of this algorithm can replace blobs of up to 2 x N where N is arbitrary. By iteratively using this dead pixel replacement method twice, blobs of up to 4 x N pixels can be replaced.

3.2.4 Column Offset

One of the most visible image defects due to non-uniformity is a vertical striping of the image. This is illustrated in some of the figures in section 3.2.8. To eliminate these non-uniformities, we use an adaptive technique of summing each column in the raw image (after dead pixel correction) and exponential averaging the resulting row in time to come up with a long term mean value for each column. We then use the long term mean value for each column to derive a long term mean value for the image. Subtracting the image mean from the column mean then creates a column offset.

$$column\_sum[j] = \frac{1}{N}\sum_i Raw[i][j] \qquad \text{where N = number of pixels in a column} \quad (3)$$

$$mean\_column\_sum = <column\_sum> \qquad \text{where the average is done over time} \quad (4)$$

$$image\_mean = mean\_column\_sum / M \qquad \text{where M = number of pixels in a row} \quad (5)$$

$$column\_offset = mean\_column\_sum - image\_mean \quad (6)$$

The column offset is then subtracted from the raw image to produce a corrected raw image that is then passed to the adaptive gain and offset correction algorithm. Correcting the raw image with this column offset is done before using the adaptive algorithm because the adaptive algorithm assumes that the surrounding neighborhood can be used to give a statistically valid estimate of the desired pixel. This assumption is not valid if there are correlated defects in the neighborhood as there would be from a defective column.

3.2.5 Low Contrast Scenes

When the image from an infrared camera is displayed, an automatic gain or histogram normalization is usually used. We have seen that the non-uniformities can get exaggerated when the scene itself has low contrast, that is, when the signal to noise ration is small. This is particularly annoying in a moving scene where the scene variations also tend to produce a jitter in the overall illumination level. Plateau equalization, described by Vickers [3], provides a good compromise solution to the display problem for low contrast scenes. Plateau equalization essentially caps the histogram values at a level that can be adaptively set for each image with the equation:

$$P_n = S(P_{n-1}) * G_{desired} / D \quad (7)$$

where ***P*** is the plateau value for the n'th image. This equation employs iterating in time, whereas, Vickers [3] iterated on the same image. Practically, they are the same for slowly moving imagery. The plateau value converges fast with most histograms. The function ***S(P)*** is the total count of pixels in the histogram given the cutoff value ***P***.

$$S(P) = \sum_{0}^{2^N} c_i \tag{8}$$

where

$$c_i = \min(C_i, P) \tag{9}$$

and $\boldsymbol{C_i}$ is the count of pixels having raw signal level, $\boldsymbol{i}$. After examining many images from the camera, particularly with low contrast scenes, we chose a value of the parameter, G, as ¼. This seemed to minimize the spatial noise. Vickers [3] preferred a value of 1 for the images that he examined. In addition, we set a minimum of 10 for the plateau value. This assured us that we would never converge to a value of 0, from which it is impossible to recover and this also assured us that we would never go all the way to a histogram projection regime, that has its own set of problems with noise emphasis.

3.2.6 Motion Requirement and Statistical Correlation of Images

Computing the desired pixel value (the quantity, *f*, in equations (2)) is a very important step in the algorithm. We have used a two-dimensional circularly symmetric sinc function (15 x 15) to provide the estimate of the desired pixel value. A sinc function can be shown to be the optimum convolution kernel for interpolation within a prescribed frequency limit. However, for the gain and offset coefficients to be valid over the whole dynamic range of the pixel, the algorithm must sample that whole range. This is accomplished best either by a moving scene or a panning camera. If a panning camera stops, then the algorithm tends to slightly wash out the image. This then creates a ghost of prominent objects in the scene when the camera resumes motion. We can eliminate this problem by detecting motion and only adapting the gain and offset coefficients when the image changes. Both these requirements are accomplished fortuitously by sampling the image at a frame rate much less that the nominal camera frame rate of 30 frames per second. If we sample at approximately one frame per second, and if the scene is changing, each pixel in the image is likely to have a much different value than it had in the previous sample. This provides the method of steepest descent algorithm good statistical coverage of the dynamic range of the independent variable and will lead to convergence. Also, if a pixel has not changed in value from the previous image (that of one second ago), then we assume that there was no motion and can choose not to adapt that pixel's gain and offset coefficients. This has the effect of stopping adaptation when there is no motion and eliminating the unwanted ghosting effect of the algorithm. We make the decision on whether a pixel has changed by comparing the desired images, not the raw images. This is done because the desired image is the result of a 15 x 15 spatial convolution and has much less temporal noise. Thus the threshold value for deciding whether a pixel has changed can be set close to the temporal noise level of the raw image and one can be confident that if the threshold is exceeded, then the pixel has truly changed. Thus it is OK to proceed with adaptation of the gain and offset coefficients.

3.2.7 Speed

The gain and offset correction must be done on every image. The dead pixel correction must be done on every image. If the image is to be displayed, the plateau equalization must be done on every image. The mean column offset must be done on every image. However, the adaptation of the gain and offset coefficients need not be done on every image. Indeed, it is wrong to perform the adaptation on every image because that does not provide the algorithm with statistically independent samples of the dynamic range of the independent variable. The plateau equalization algorithm requires a histogram and look-up table creation. If the scene is changing slowly, the histogram and creation of the look-up table need only

be done once every N frames. We have been conservative and used N = 2. The display of the image every frame requires the image to be passed through the look-up table every frame. These requirements are summarized in Table 2.

Table 2. Functional Processing Requirements

Operation	Every Frame	Every 2 Frames	Every 30 Frames
Gain Multiplication	x		
Offset Addition	x		
Dead Pixel Correction	x		
Mean Column Offset	x		
Image Display	x		
Histogram Calculation		x	
Equalization Look-up Table Creation		x	
Gain and Offset Adaptation			x

3.2.8 Temperature Drift, Dynamic Range and Long Term Stability

A fixed gain and offset corrected image can look quite good, particularly at one of the two temperature points used for calibration. Figure 5 illustrates the results of a two-temperature calibration. The two temperatures used were approximately 4.4 ^{o}C and 21.7 ^{o}C. Dead pixels are shown as completely black. This is done so that subsequent subjective comparisons of the imagery in this paper are not confused by differences in dead pixel replacement strategies. Figure 5 shows the output of the Boeing camera with a fixed gain and offset at the upper temperature calibration point 21.7 ^{o}C. The figure shows a coffee cup in front of a disk drive on a lab bench. A diagonal cable is also stretched across the scene. Panning a camera back and forth automatically across the lab bench generated the images in this paper. A picture of the lab bench taken in the visible spectrum is shown in Figure 6.

Figure 5. Two Point Temperature Corrected Image at T = 21.7 ^{o}C

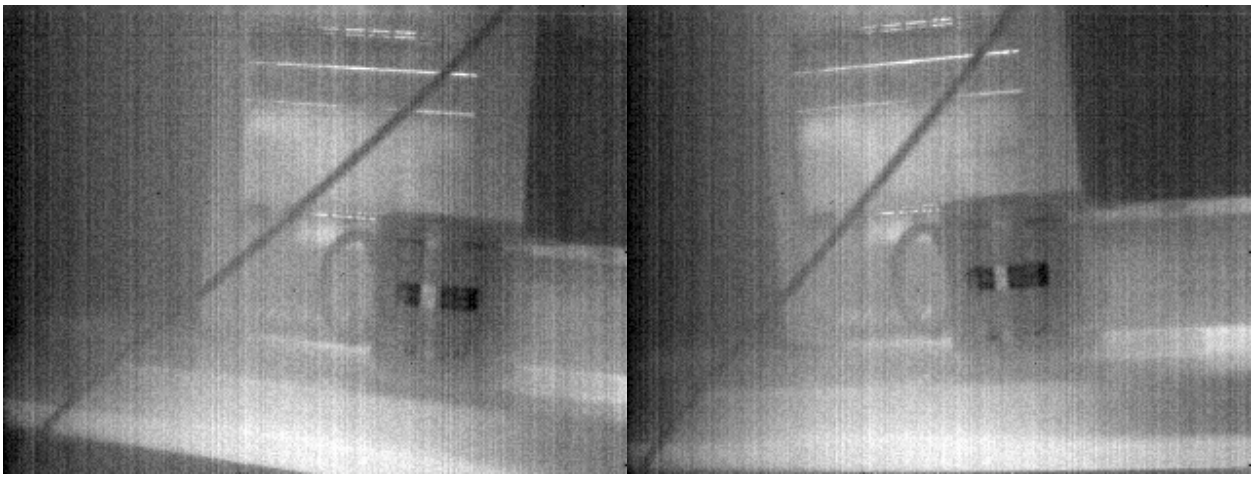
The adaptive gain and offset correction algorithm also converges to a good set of coefficients as illustrated in Figure 7. Initially, the image at Frame = 1 has a gain of 1.0 and an offset of 0. At frame

numbers 1, 100, 1,000, 10,000, 50,000 and 100,000, the panning of the camera was stopped and the same scene with the coffee cup placed in front of a disk drive enclosure was saved to disk. Mildly warm air is being vented by the disk drive to warm the coffee cup to a temperature of 26.6 ^{o}C =. The scenes observed by the camera vary from low contrast imagery to extremely low contrast imagery. This imagery was chosen because gain and offset errors are very visible in such scenes. Errors show up mainly as vertical stripes and individual pixel errors. The gain and offset adaptation was done every 30 frames. After 1000 frames, most of the offset errors are gone and after 10,000 frames, the coefficients are mostly converged and at 50,000 frames, they are fully converged.

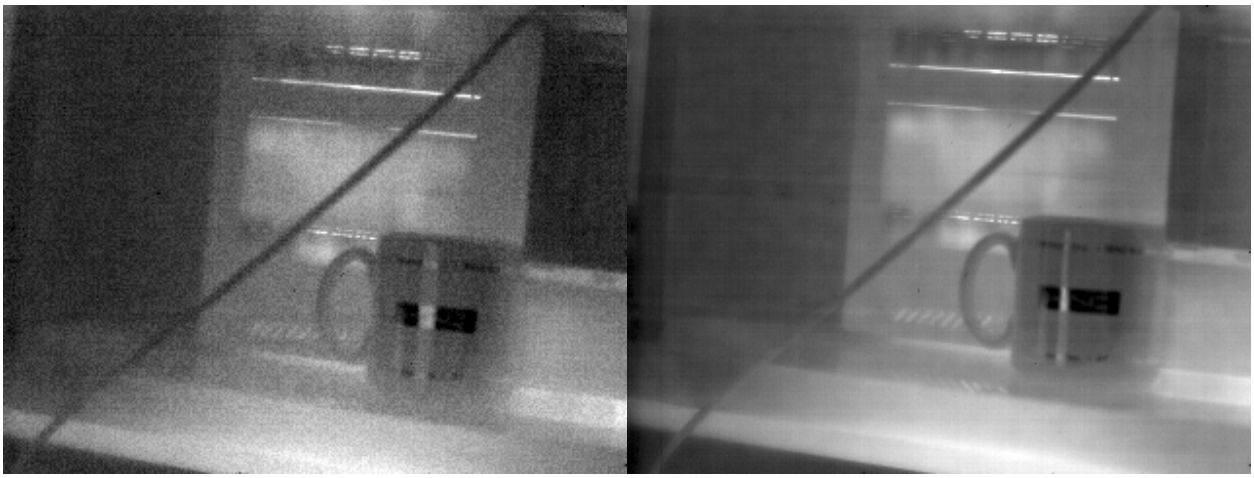
Figure 6. Visible Spectrum Image of Objects on the Lab Bench

Figure 8 shows the same images used in Figure 7 using the fixed gain and offset coefficients from a full two-point temperature correction. The ambient temperature in the room varied approximately between 21 ^{o}C and 22 ^{o}C. As the ambient temperature in the room changed, the operating point of the sensor changed and the fixed gain and offset coefficients became no longer valid. The images in Figure 8 have significant errors while the images shown in Figure 7 that were produced using adaptive gain and offset coefficients remains good as the temperature varies.

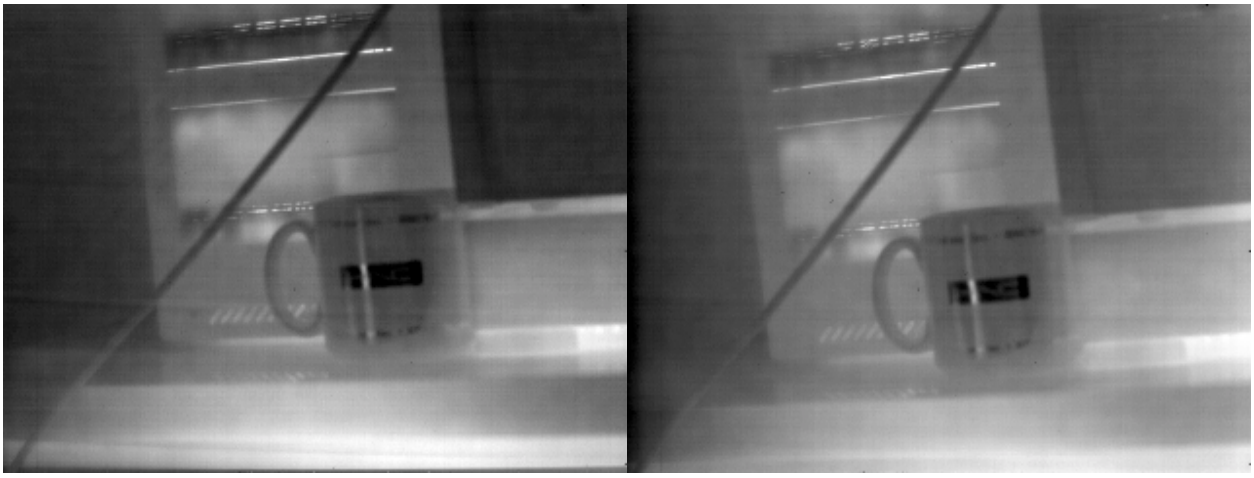
The ability of the adaptive gain and offset algorithm to compensate for temperature variations becomes even more important as the temperature varies over a wider range. Most of the infrared photons (approximately 80%) that are sensed by the microbolometer actually come from the lens body. Thus a relatively small ambient temperature increase of three degrees Celsius causes many pixels to approach saturation. The adaptive gain and offset algorithm is able to correct somewhat for this effect as seen in Figure 7 where the temperature varied by about one ^{o}C. The uncooled infrared bolometer sensors have a very large dynamic range. The A/D converter that transforms the signal from the sensor into a 12-bit value is arbitrarily placed in that range so that the mean flux gives a value of about ¼ of the maximum and the gain is set so that small temperature changes may be distinguished. The placement of this A/D window into the whole scene can be controlled in the Boeing camera by a signal, V_{reset}. We made use of this signal to extend the dynamic range of the camera. When V_{reset} is modified, the sensor is in a new regime and any fixed gain and offset coefficients are no longer valid. This is illustrated in Figure 9 where the ambient air temperature and lens body temperature vary over a 6 ^{o}C range. V_{reset} is manually set to increasing values between 110 and 118 over this same range. If we did not control and modify V_{reset} over this temperature range, then the whole image would have saturated completely.



Frame = 1 Frame = 100

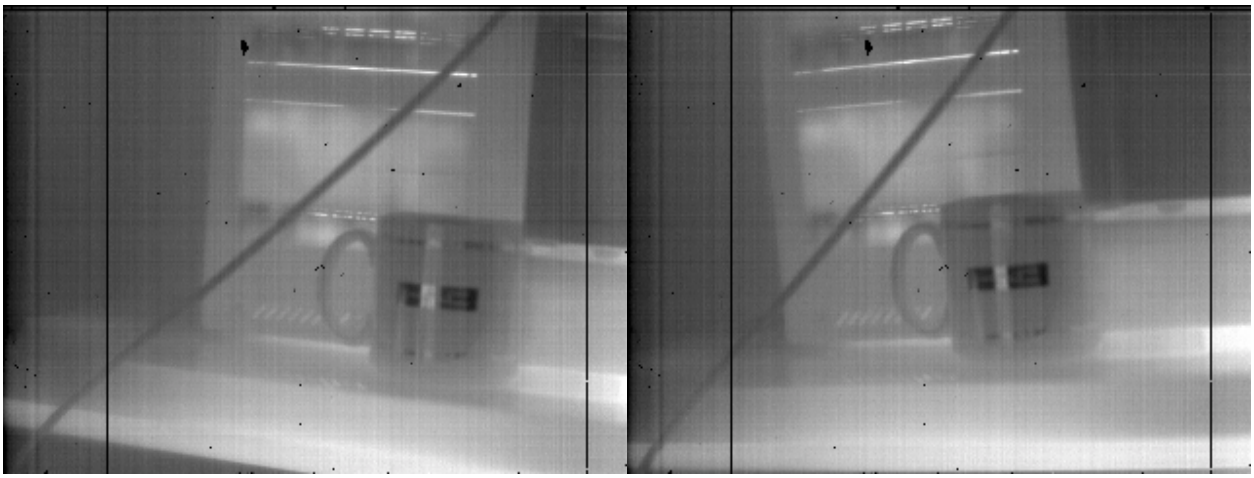


Frame = 1,000 Frame = 10,000

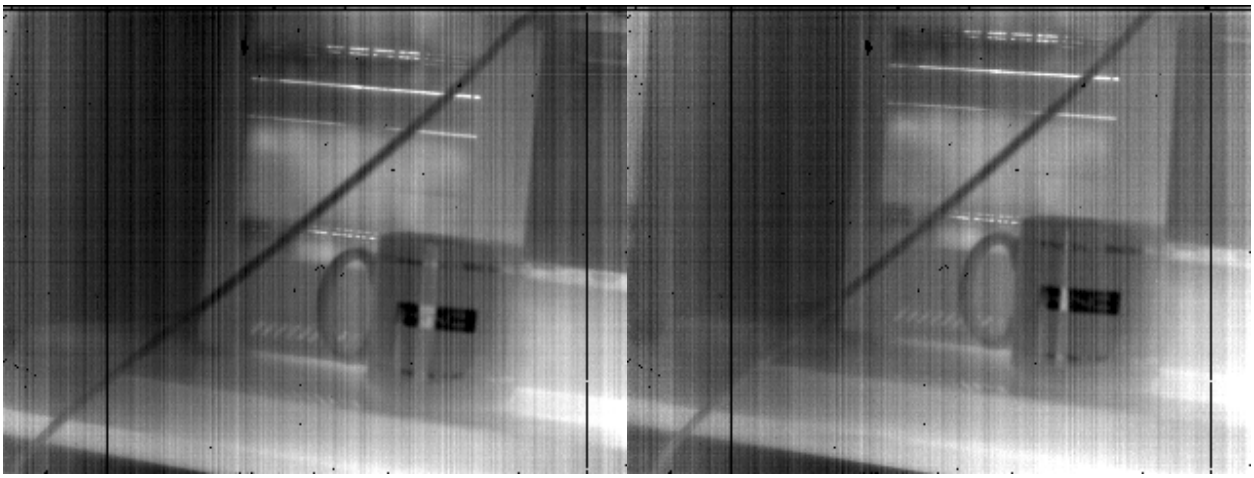


Frame = 50,000 Frame = 100,000

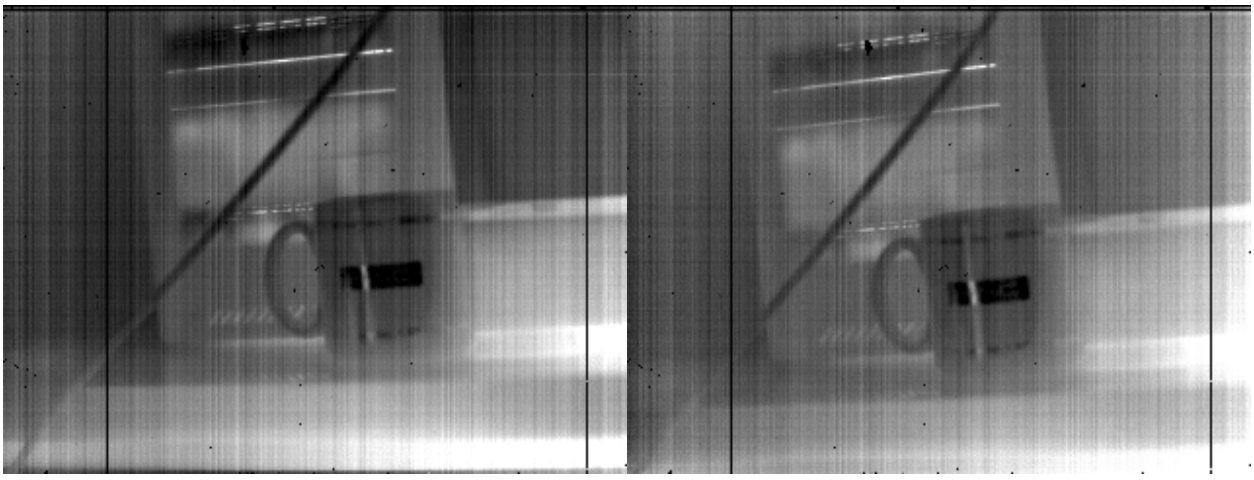
Figure 7. Images Processed with Adaptive Gain and Offset Algorithm



Frame = 1 Frame = 100



Frame = 1000 Frame = 10,000



Frame = 50,000 Frame = 100,000

Figure 8. Images Processed with Fixed Gain and Offset Correction

Figure 9 compares the fixed gain and offset image quality with that of the adaptive gain and offset image quality as a function of ambient temperature and V_{reset}. Even a small increase in temperature degrades the quality of the image generated using fixed gain and offset coefficients. As the temperature increases, the image quality of the fixed gain and offset image remains in a degraded state while the quality of the adaptive gain and offset image remains relatively constant.

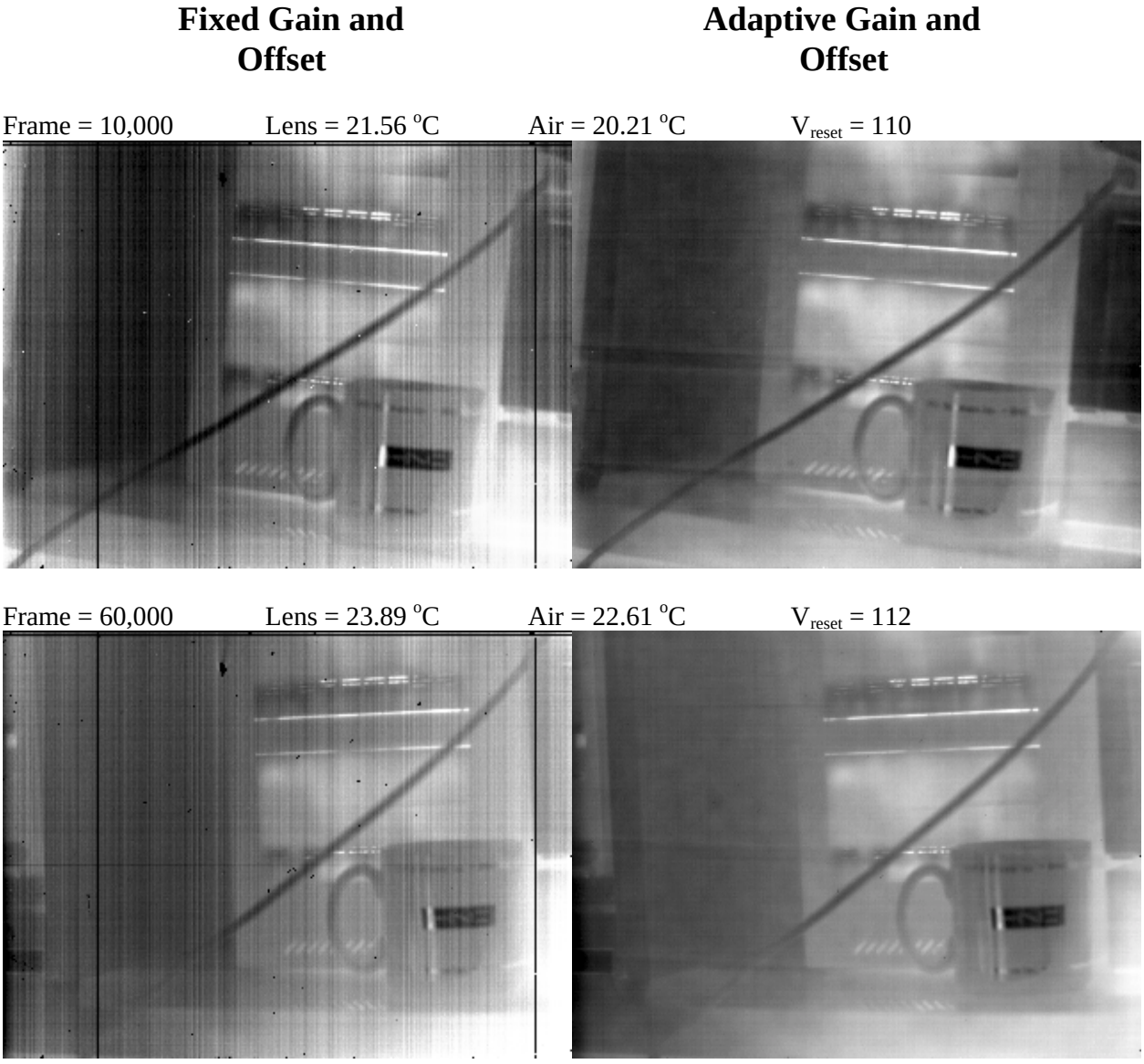
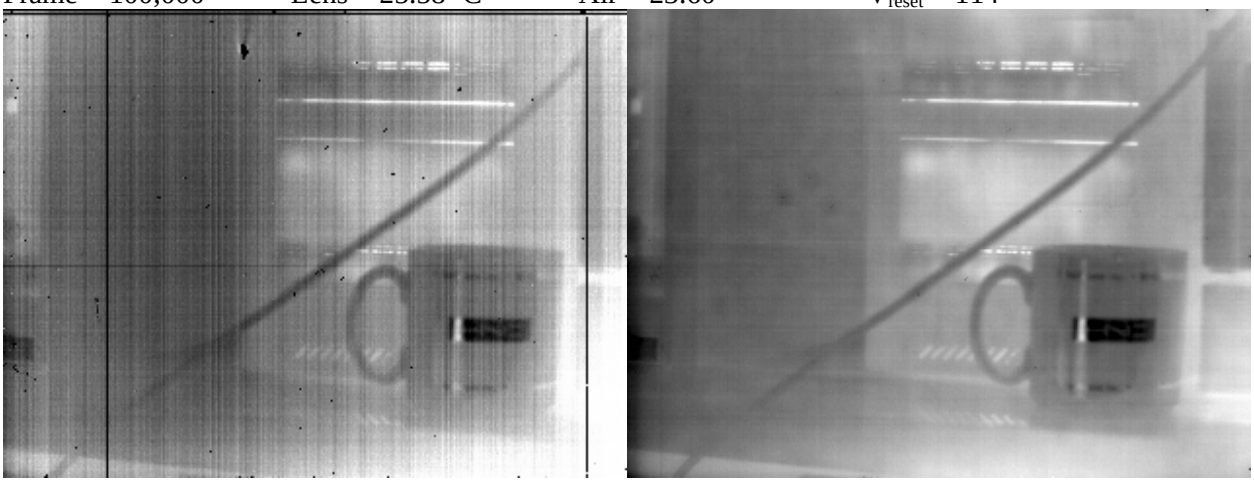
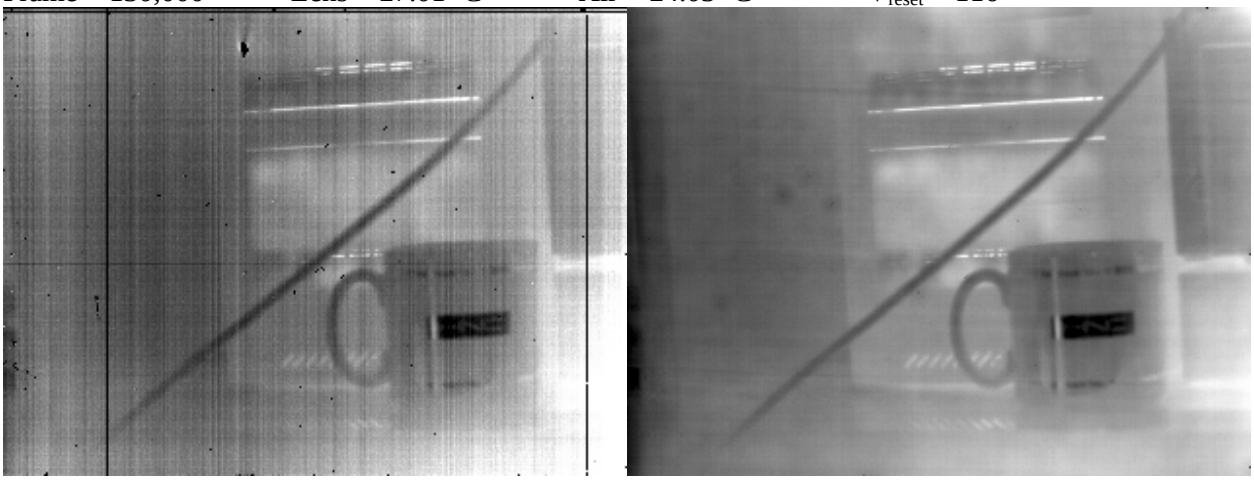


Figure 9. Comparison of Image Quality as a Function of Ambient Temperature and V_{reset}

Frame = 100,000 Lens = 25.58 °C Air = 23.60 V_{reset} = 114



Frame = 130,000 Lens = 27.01 °C Air = 24.03 °C V_{reset} = 116



Frame = 160,000 Lens = 28.32 °C Air = 25.26 °C V_{reset} = 118

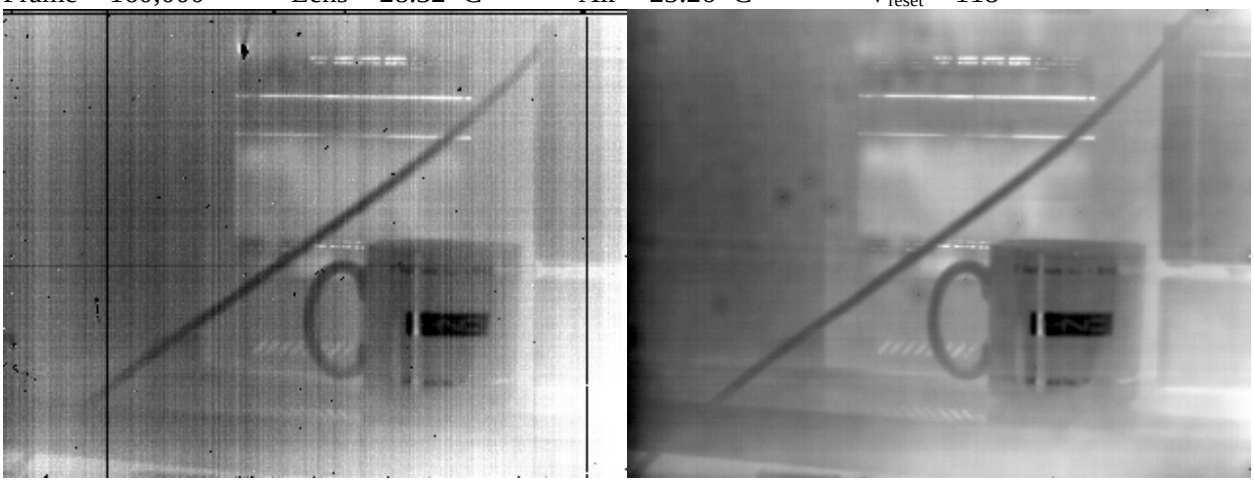


Figure 9 (continued). Comparison of Image Quality as a Function of Ambient Temperature and V_{reset}

4.0 CONCLUSIONS AND RECOMMENDATIONS

We have applied several techniques to solve the non-uniformity correction problem applied to images from an uncooled bolometer camera. The real time implementation allows long term stability and performance issues of the algorithm to be addressed. We determined that the adaptive algorithm works better when the time sample between images of a moving scene is large; that is, when the images are relatively uncorrelated. This effect must be balanced by the need to have the algorithm converge in a finite time. The net effect of this balance is that the hardware signal processing requirements are reduced considerably since many algorithmic calculations need not be done on every frame. The neural network, by itself, is not adequate. We have also had to 1) replace dead pixels accurately, 2) use a column offset to eliminate correlated column defects, 3) employ plateau equalization to create a pleasing output image, and 4) adapt the A/D converter voltage offset as the temperature changes to keep the mean value of the image within the dynamic range of the A/D converter. The net result of the signal processing yields a camera that can generate good imagery over a much larger dynamic ranges than a camera without such signal processing.

5.0 ACKNOWLEDGEMENT

This work was supported by the Naval Research Laboratory under contract N00014-96-C-2061.

6.0 REFERENCES

[1] D. A. Scribner, S. Michaels, and M. R. Kruer, “Adaptive Nonuniformity Correction for IR Focal Plane Arrays Implemented on a High Speed Parallel Processor”, IRIS Specialty Group on Passive Sensors, Feb. 1996.

[2] B. Widrow and S. Sterns, Adaptive Signal Processing, (Prentice-Hall, New Jersey, 1985).

[3] Virgil E. Vickers, “Plateau Equalization Algorithm for Real-time Display of High Quality Infrared Imagery”, Opt. Eng., **35**(7), pp.1921-1926, July 1996.